1 Objectives

1.1 Instructor

In this lecture we want to make the students familiar with record types and disjoint types.

1.2 Student

- ...

1.3 Maturation

2 Outline

First ask the question, "why do we have user-defined types?". The answers will probably involve abstraction, ease of programming, etc.

Show a slide that covers the two types. Sometimes we want to collect a group of data elements as one element (a *record*), sometimes we want to have a collection of different kinds of items that are the same type (a *disjoint type*).

Remind the students about tuple notation. Then show them the code to handle complex numbers, using tuples. Ask them what is wrong with writing code that way. They should answer that it's hard to manage, doesn't maintain abstraction, those kinds of things.

Now show them the syntax for a record definition. Show them the code to perform complex addition, and show them a few examples. As an activity, make them write the function to do complex multiplication. Have a student read the code as you type it in and test it.

The more complicated syntax, such as `with`, should probably be left for an MP.

Next, explain to them the idea of a disjoint data-type.

Show them how to define a Binary Tree. Write the `find` function first, and show them the code running on a few examples.

Next, show them add. This will be complex, because it creates a new structure, it does not replace the old one, which is what many of them may expect. Show them what happens if they do several `add` operations in a row, and make sure they are familiar with the concept that a change propagates from the change point to the root of the data-structure.

Next, have them write delete. This should be done "live" if possible, but have the slides ready just in case. There are three cases to consider; first have them code up the test to see which case applies. Next have them code the "no child" case, then the "one child" case, then the "two child" case. The one and two child cases will have symmetries.

Show them the Option type. Motivate the type by asking what `find` should return if the value is not there.

Now show them linked lists, with Cons and Nil. Have them write the type, and the functions.

3 Resources